Electronic Version
Stylesheet Version v1.1.1

Description

Automatic Configuration of Network Automation Devices

Cross Reference to Related Applications

[0001] This application is related to US Patent Application number 09/635,280, filed August 9, 2000, entitled "A Method and Apparatus for Programming a Factory Automation Device", herein incorporated by reference. It is also related to US Patent Application number 09/973,068, filed October 10, 2001 entitled "Method of Configuring an Automation Module on a TCP/IP Network" (herein incorporated by reference), which is based upon French Patent Application 00 13 191 filed October 12, 2000.

Background of Invention

[0002] Field of Invention. This application is in the field of factory and industrial automation, particularly in the area of networking in factory automation.

[0003] In the field of factory automation, the continuous operation of the factory is of critical importance to the opera-

tors of the factory. Vendors of factory automation equipment design their equipment to allow for non stop, maintenance free operation in harsh environments. Equipment is designed to handle extended temperature ranges, excessive vibration and shock, and sometimes for contaminants in the air. In spite of the best of efforts by automation vendors, the harshness of factory environments sometimes causes equipment to fail.

[0004] Since many factories operate 24 hours a day, 7 days a week, it is very important to return the factory to an operating state as soon as possible. Factory downtime can cost the factory operators millions of dollars a day for downtime. Failures also occur at random times, and it is not uncommon for a failure to occur on a second, third or weekend shift, when the maintenance staff is limited, and the plant engineers are unlikely to be working.

[0005] As a result, automation vendors search for means for making the replacement of failed units as simple as possible. This allows untrained operators or factory maintenance staff to replace a failed unit with the minimum of training or instruction.

[0006] Factory automation equipment, such as programmable logic controllers (PLCs), IO modules (such as analog input,

digital input, analog output, or digital output modules), motion controllers, vision controllers, invertors, encoders, process controllers, numerical controllers, relays, sensors, bar code readers, weighing stations, cubing machines, power monitoring equipment, breakers, industrial power monitors, and a number of other devices are now connected to networks and are becoming increasingly intelligent. As a result, steps need to be taken to configure or program both the device and its network parameters. Traditionally, this requires a plant engineer to program the device and a network engineer to set the network parameters, especially on complex networks such as Ethernet.

[0007] However, this complexity of individual devices is in direct conflict with the goal of simplicity in the replacement of failed units. This invention is designed to overcome this conflict by allowing for the easy replacement of factory automation devices as well as to solve other problems in factory automation.

[0008]

Summary of Invention

[0009] This invention describes a method of automatically configuring an automation device, which could be a pro-

grammable logic controller, an IO module, or any another automation device, operating on an automation specific protocol, such as MODBUS/TCP, MODBUS, or another protocol, the method including the steps of searching for the address of a configuration server, searching the memory of the configuration server once it is found, and loading the configuration from the configuration server to the automation device over the automation specific protocol.

[0010] This invention further describes a factory automation system that automatically configures automation devices, which could be a programmable logic controller, an IO module, or any another automation device, on an automation specific network, such as MODBUS/TCP, MODBUS, or another protocol, wherein the system includes automation devices that are designed to search for a configuration server and then search to find specific configurations within the configuration server, where the configuration server contains a linked list of configurations that are available to the automation devices.

[0011] This invention also describes an automation device, which could be a programmable logic controller, an IO module, or any another automation device, that is connected to an automation specific network, such as MODBUS/TCP,

MODBUS, or another protocol, where the automation device is designed to search the network for a configuration server, then search the configuration server for its specific configuration, and then load the configuration from the configuration server into the automation device.

[0012] This invention also describes the replacement of a first automation device with a second automation device on the automation specific network, wherein the identifier for the first device is replaced with the identifier of the second within the configuration server, and the second automation device is designed to search the network for a configuration server, then search the configuration server for its specific configuration, and then load the configuration from the configuration server into the automation device.

[0013] Other systems, methods, features, and advantages of the present invention will be, or will become, apparent to one having ordinary skill in the art upon examination of the following drawings and detailed description. It is intended that all such additional systems, methods, features, and advantages be included within this description, be within the scope of the present invention, and be protected by the accompanying claims.

Brief Description of Drawings

[0014] The invention can be better understood with reference to the following drawings. The components in the drawings are not necessarily to scale, emphasis instead being placed upon clearly illustrating the principles of the present invention. Moreover, in the drawings, like reference numerals designate corresponding parts throughout the several views.

[0015] Figure 1 is a simplified block diagram showing a factory network;

[0016] Figure 2 is a flowchart demonstrating a possible startup sequence for a factory automation device utilizing the invention; and

[0017] Figure 3 is an example of a memory map of a configuration server's memory in accordance with the present invention.

[0018]

Detailed Description

[0019] While this invention is susceptible of embodiments in many different forms, there is shown in the drawings and will herein be described in detail preferred embodiments of the invention with the understanding that the present

disclosure is to be considered as an exemplification of the principles of the invention and is not intended to limit the broad aspects of the invention to the embodiments illustrated.

[0020] Figure 1 is a simplified diagram of a factory network system. It contains a network 100 that connects automation devices 103, 104 and 105, and a configuration server 101. Other devices and servers can also be connected to the network 100. Moreover the configuration server 101 may perform other functions in addition to those described here.

[0021] The network 100 uses an automation specific protocol such as MODBUS, MODBUS/TCP, Devicenet, Profinet, CANOpen, Ethernet/IP, Fieldbus Foundation, or other protocol designed specifically for use in automation.

[0022] The MODBUS and MODBUS/TCP protocols are defined in the MODBUS/IDA submission to the IEC (Commission Electrotechnique Internationale, www.iec.ch), herein incorporated by reference. The MODBUS/TCP protocol is a set of protocols that operate on Ethernet, and are often operating over the Internet or over an Intranet. As a result, the connectivity provided by network 100 does not need to be local, but instead could connect devices 103, 104, and

105 and server 101 that are thousands of miles apart.

[0023] Devices 103, 104, and 105 are factory automation devices that have network connectivity and intelligence. Because of the intelligence, these devices can be configured both for their network parameters and for their operation within the factory environment. However, the configuration of these parameters are often specific to the particular function that the particular device is doing in the specific location in the plant. It would typically not be desirable for the configuration of device 103 to be placed in device 104. However, in some circumstances, this may be required. With potentially thousands of factory automation devices in a factory, the management of the configurations for each device is quite difficult. Should a device 103 fail, then the device needs to be replaced with the same type of device, and the same configuration needs to be loaded into the device. When first installed, a plant engineer does the programming and make sure that the new configuration is tested. Should a replacement be required on an off shift, the plant engineer may not be available. In this case, the configuration needs to be downloaded with the minimum of training. Since a type of device is often used to provide many different types of functionality, it

can not be stored pre-configured in the tool crib.

[0024] As a result, when the replaced device 103 is connected to the network 100, it needs a specific configuration to perform the function that the original device was performing. The device 103 requests a download from the Configuration Server 101 when it is replaced. This process is further discussed below in the discussion of Figure 2.

[0025] The configuration server 101 has an attached bar code reader 102. The configuration server 101 could be a personal computer or a programmable logic controller or any other device with the storage capability to store the configurations for the devices 103, 104, and 105. The configuration server 101 is also connected to the network 100.

[0026] In one possible embodiment, the network 100 is a MODBUS/TCP network. The configuration server 101 is a programmable logic controller or a personal computer emulating a programmable logic controller's memory map. The memory of a programmable logic controller using a MODBUS type protocol has an address range from 40000 to 46535 words. The configuration is downloaded to the devices 103, 104 or 105 from that memory. Further discussion of the memory map can be found below in the discussion of Figure 3.

[0027] Connected to the configuration server 101 is bar code reader 102. When one of devices 103, 104 or 105 fails, the operator removes the device, tells the configuration server 101 that a replacement will occur, scans a bar code on the device 103, 104, or 105 with the bar code reader 102, and then scans the bar code of the replacement unit. The bar code on each device contains an identifier for the device, such as the unique MAC network address. The new device 103, 104, or 105 is then connected to the network 100 in the same way the failed unit was connected. The new device 103, 104, 105 then searches and requests a download of its configuration from the configuration server 101. Further information concerning the use of bar codes and bar code readers for the replacement of automation devices can be found in U.S. Patent Application number 10/707,482 "Bar coded addressing technique", herein incorporated by reference.

[0028] While the above description discusses the replacement of a failed device 103, the techniques described here could also be used to restart a network 100 of devices 103, 104, 105 after a power failure, or in the configuration of a new device 103, 104, 105 where the configuration is stored in the configuration server 101 by a programming tool.

[0029] Figure 2 is a flow chart of the operation of the devices 103(or 104 or 105) upon power up. After the device 103 is turned on, it executes a normal power up sequence 200. This sequence will initialize the device and will start the basic communications stack for accessing the network 100.

[0030] The next step involves finding the address 201 of the device 103. There are a number of schemes available for determining the address of the device, as seen in the references that are included in this application. The simplest mechanism on an Ethernet network is to have the bar code reader 102 scan the old MAC address and the new MAC address, and replace the address in the DHCP or BOOTP tables. These tables are stored in the configuration server 101. As the device 103 starts up the protocol stacks, it sends out a BOOTP/DHCP message to ask for its IP address. The response to this message will provide the device 103 with its address, allowing the stack to complete its operation. For other types of networks, this may involve setting of thumb wheels on the device itself or connecting via a known address to set the new address.

[0031] Once the device 103 has found its address, the next step that it needs to perform is to find the address of the con-

figuration server 101. This is first done by searching 202 the network for the addresses of devices that are available on the network 100. This search 202 can use one of several schemes. The search 202 could start with address 1 on the local sub network and linearly search from xx.xx.xx.01 to xx.xx.xx.255. Alternatively, a search could use the MAC addresses of itself, and search linearly above and below that address on the assumption that the server was manufactured by the same vendor (and therefore has a similar MAC address). Another scheme involves reading the ARP tables on a local machine and searching 202 each address in that table for the server. A read memory message is then sent to each server to see the address is still present on the network and to see if the device at that address can respond using the automation specific protocol. In the case of our example, to see if the device at that address responds on IP port 502, assigned to MODBUS/TCP.

[0032] If no address is found 203, the device 103 goes into a delay 204 for 30 seconds or so, and restarts the search 202. This restart is necessary for the case where the configuration server powers up more slowly than the device, and becomes available later.

[0033] If the address is found 203, then the unit at that address

is searched to see if the address is a configuration server 101. The search is best done by reading memory 300 to see if there is a linked list present that contains device types and device IDs. The memory map 300 in the server 101 is defined in Figure 3 and described later in this document. However, the configuration server itself could perform the memory search in an alternative implementation. In this alternative, the automation device 103 would provide the configuration server 101 with the Device ID, and then the configuration server would download the configuration 306 to the automation device 103 if the configuration for that particular automation device 103 is found.

[0034] Next, the memory 300 of the configuration server 101 is searched 205 to see if there are any records of the same device type as the device 103. Given that the configuration server is a linked list of configuration records, this process is a search of the linked list for the given device type.

[0035] If no records are found that are the same 206 as this device 103, then the device 103 will continue searching for valid addresses 202. It is possible that there are multiple configuration servers 101 on a network 100, so each may have to be searched to find the correct configuration

record.

[0036] If the device types do match, then the next step is to search for the Device ID 207. This search finds the exact configuration for the device 103. Although a number of schemes could be used to match devices, for an Ethernet network, the MAC address is perhaps the most unique number that is readily available. MAC addresses are assigned as unique blocks to manufactures to assign to individual products and are unique to each product that is manufactured. Serial numbers could also be used, as could IP addresses or URLs. This search is down a linked list of configuration records and involves a simple comparison of the device's 103 MAC address to the individual marker 304 in memory 300.

[0037] If the address is not found at step 208, then the search for device type resumes at 205.

[0038] If a match at step 208 has been found, then the search has been successful. The configuration is now downloaded 209 to the new device 103. The term configuration is broad, and could be as simple as a byte or two of information, for either the network interface, the device, a sub-device, or any combination thereof, or could involved the download of a set of multiple programs, parameters,

and data for a set of modules.

[0039] The configuration download 209 occurs after first verifying that the correct record has been found. A CRC-16 check is done to assure that a valid configuration has been found, either before or after the configuration is sent from the configuration server 101 to the device 103. The device's MAC address has already been checked in the search 207, and additional information may be checked to increase the assurance that this is a valid configuration record for this particular device 103. The configuration is downloaded by the device 103 reading the memory of the configuration server 101 at the location of the found configuration.

[0040] Once the configuration download 209 has completed and has been verified, the device 103 starts operating 210 using the downloaded configuration.

[0041] Figure 3 shows a possible memory map 300 of the configuration server 101. The example here is of a unit that uses the memory addressing scheme dictated by the MODBUS protocol. This addressing scheme utilizes a memory bank 65535 words long starting at 400001.

[0042] The memory is organized as a linked list starting at 400001 with a pointer 301 to the linked list. This pointer

points to a device marker 302.

[0043] The device marker 302 identifies the type of device configuration records that are stored at this location. It is used in the search for device type 205 step. This marker could contain a numeric value for the part number of the device or a string that contains the product name of the device.

[0044] After the device marker 302 is a pointer 303 to the next device type 310 in the linked list. If this is the last device type in the list, a null pointer is entered, typically a zero. In Figure 3, this is the case for pointer 311.

[0045] After the pointer 303 is the first individual marker 304 for a specific device. The individual marker 304 could be a MAC address, and is used in step 207 in the search for the specific device.

[0046] The individual marker 304 is followed by a pointer 305 to the next individual marker 307 in the linked list. This next individual marker 307 also has a pointer 308 to the next record. Should the pointer 308 be null, then this is the end of the list of configuration records for this device type 302.

[0047] After the individual marker 304 and its pointer 305, we have the configuration 306 that is downloaded to device

103 if the device marker 302 matches its device type and if its individual marker 303 matches its MAC address.

[0048] The configurations 309 and 314 contain the configurations for other devices, as identified by their individual markers 307 and 312.

[0049] The memory structure 300 containing the linked list of configurations is created with a utility software that may run on the configuration server 101 or on another system. This software may upload configurations from devices to archive them in the configuration server or may be part of the programming software that is used to program the device 103, 104, and 105.

[0050] It should be emphasized that the above-described embodiments of the present invention, particularly, any preferred embodiments, are merely possible examples of implementations, merely setting forth for a clear understanding of the principles of the invention. Many variations and modifications may be made to the above-described embodiment(s) of the invention without substantially departing from the spirit and principles of the invention. All such modifications are intended to be included herein within the scope of this disclosure and the present invention and protected by the following claims.